IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Gunter SCHWARZBAUER ET AL.

Confirmation No.: 9707

Application No.: 10/676,227

Art Unit: 2176

Filed: September 30, 2003

Examiner: Amelia L. RUTLEDGE

For: **AUTOMATIC CONTEXT MANAGEMENT FOR WEB APPLICATIONS WITH CLIENT SIDE CODE EXECUTION**

APPELLANTS' BRIEF ON APPEAL UNDER 37 C.F.R. § 41.37

MS Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Appellants submit this Brief in accordance with 37 C.F.R. § 41.37 in support of their appeal from the Final Office Action, mailed July 14, 2006, by Examiner Amelia L. Rutledge, and the Advisory Action, mailed November 14, 2006, in the above-identified patent application.

In accordance with 37 C.F.R. §§ 41.31 and 41.37, this brief follows the January 16, 2007 filing of a Notice of Appeal and payment of the required fee. This brief is in support of said Notice of Appeal. Concurrently, herewith, Appellants submit a Petition for Extension of Time Pursuant to 37 C.F.R. § 1.136(a) requesting an extension of 3 months in which to submit this Appeal Brief. The requisite fee under 37 C.F.R. § 1.17(a) is also enclosed. With such petitions and fee, this Appeal Brief is due on or before June 18, 2007, and thus, is timely. (The deadline with a three-month extension is Saturday, June 16, 2007, and thus, the actual deadline is Monday, June 18, 2007.) Appellants submit that no further extension of time fee is required. However, the Commissioner is

hereby authorized to charge any unpaid fees deemed required in connection with this Appeal Brief, or to credit any overpayment, to Deposit Account No. 04-0100.

I. REAL PARTY IN INTEREST

The real party in interest for this appeal is Borland Software Corporation. The inventors having assigned their rights in and to this application to Borland Software Corporation, such assignment having been duly recorded.

II. RELATED APPEALS AND INTERFERENCES

To the Appellants' knowledge, there are no other appeals, interferences, or judicial proceedings which will directly affect, be directly affected by, or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-31 are pending in the application.

This appeal is in respect of the rejection of claims 1-31.

There are 31 claims pending in the application, *i.e.*, claims 1-31. They are reproduced in the **Claims Appendix**. The current status of the application's claims is as follows:

1. Claims canceled: none;

2. Claims withdrawn from consideration but not canceled: none;

3. Claims pending: 1-31;

4. Claims allowed: none;

5. Claims rejected: 1-31.

IV. STATUS OF AMENDMENTS

Appellants filed an Amendment in Response to Final Office Action on November 28, 2006, amending claims 1, 9-11, and 17-27. The Examiner indicated in the November 14, 2006 Advisory Action that for the purposes of appeal, the amendments would be entered.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The present invention relates to a system and a method for testing, monitoring, and automating network applications having client-side executable code (e.g., JavaScript, DHTML, Java Applets, or ASP). The invention can receive standard web documents containing forms and client-side executable code from a web server and respond by recording and replaying context-full scripts to simulate a user or client. Thus, multiple connections to a server can be simulated to test the performance of the server under varying user loads.

Web pages that are transmitted to the client can include HTML forms which allow a user to input information which is transmitted to the server, where it can be used to determine the server's response. Web pages transmitted by the server can also include client-side executable code, which, when processed by the client, can generate HTML and HTML forms. Execution of client-side executable code can be time-consuming and resource intensive. Further, client-side executable code can dynamically produce unique web links (i.e., URLs) which have not been previously recorded by the client. The present invention can parse client-side executable code and incorporate appropriate instructions into the context-full script (including dynamic and unique URLs) to generate a meaningful response without executing the client-side executable code.

The present invention is directed to "[a] system for automatic context management for testing, monitoring and automating a network application having client side executable code." For example, *see* claim 1, preamble. Independent claim 1 recites "a tool operable to parse said client side executable code," (Specification, page 17, lines 13-16; Figure 11, item 1111), "so as to determine a subsequent state of the network application free of interaction with a user." (Specification, page 4, line 21 - page 5, lines 1; *see generally* Figure 6). Claim 1 further recites "a recorder operable to record at least one context-full test script; and" (Specification, page 16, line 22

- page 17, line 3; *see e.g.*, Specification, page 24, lines 15-18) "a replay engine of said tool having a context-full API, said replay engine operable to execute said context-full test script." (Specification, page 30, lines 9-11; Specification, page 18, lines 5-9).

Independent method claim 28 is directed to "[a] method of fuzzy form detection." (Specification, page 19, 7-15). Claim 28 recites the steps of "comparing a form to be submitted to at least one form in a session history;" (Specification, page 15, 20-23; Specification, page 16, lines 3-12) "generating data based upon differences resulting from the comparing step;" (Specification, page 25, line 21 bridging page 26, line 5) "performing said comparing and generating steps for each form in said session history;" (Specification, page 26, lines 6-9) "choosing one of the forms in said session history having the greatest similarity to said form to be submitted based upon the generating step results; and" (Specification, page 26, lines 10-16) "applying form merging instructions to said chosen session history form to obtain a resulting form that is substantially identical to said form to be submitted." (Specification, page 26, lines 17-21; *see generally* Figure 8).

Independent claim 29 is directed to "[a] device for automatic context management for testing, monitoring and automating a network application having client side executable code." Claim 29 recites "a processor;" (Specification, page 2, lines 2-4; Specification, page 24, lines 1-2) "a memory storing processing instructions for controlling the processor," (Specification, page 24, lines 1-2) by which the processor is configured to "record at least one context-full test script; and" (Specification, page 16, line 22 - page 17, line 3; *see e.g.*, Specification, page 24, lines 15-18) "execute said context-full test script using a context-full API for a replay engine of a tool" (Specification, page 30, lines 9-11; Specification, page 18, lines 5-9) "said tool operable to parse said client side executable code" (Specification, page 17, lines 13-16; Figure 11, item 1111) "so as to determine a subsequent state of the network application free of interaction with a user." (Specification, page 17, lines 13-16; Figure 11, item 1111).

Independent claim 30 is also directed to a "[a] device for automatic context management for testing, monitoring and automating a network application having client side executable code." The device includes "a processor;" and (Specification, page 2, lines 2-4; Specification, page 24, lines 1-2) "a memory storing processing instructions for controlling the processor," (Specification, page 24,

lines 1-2). The processor is operative to "record at least one context-full test script;" (Specification, page 16, line 22 - page 17, line 3; *see e.g.*, Specification, page 24, lines 15-18) "determine at least one parser extension using an extensible document parser" (Specification, page 17, lines 17-20) "of a context-full page-level API for a replay engine of a tool," (Specification, page 17, lines 4-8) "said tool operable to parse said client side executable code" (Specification, page 17, lines 13-16; Figure 11, item 1111) "so as to determine a subsequent state of the network application free of interaction with a user;" (Specification, page 4, line 21 - page 5, lines 1; *see generally* Figure 6) "include a replay instruction specifying at least one parser addition and parameters for said parser addition in said at least one parser extension;" (Specification, page 30, lines 9-11) "select said parser addition for said parser extension from a library of parser additions;" (Specification, page 23, lines 9-11) "implement a specific parsing algorithm using each of said parser additions of said library of parser additions;" (Specification, page 22, line 22 bridging page 23, line 1) "implement an algorithm for parsing hyperlinks from a HTML document using said parser addition of said library of parser additions;" (Specification, page 21, lines 11-14) "parse hyperlinks by searching text between a left and a right boundary string using said algorithm;" (Specification, page 22, lines 3-8) "and execute said context-full test script using said context-full page-level API for said replay engine." (Specification, page 30, lines 9-11; Specification, page 18, lines 5-9)

Independent claim 31 is directed to "[a] device for automatic context management for testing, monitoring and automating a network application having client side executable code." The device comprises "a processor;" (Specification, page 2, lines 2-4; Specification, page 24, lines 1-2) "and a memory storing processing instructions for controlling the processor," (Specification, page 24, lines 1-2) by which the processor is configured to "record at least one context-full test script;" (Specification, page 16, line 22 - page 17, line 3; *see e.g.*, Specification, page 24, lines 15-18) "generate form merging replay instructions" (Specification, page 26, lines 17-21) "in a context-full page-level API for a replay engine of a tool," (Specification, page 30, lines 9-11; Specification, page 18, lines 5-9) "said tool operable to parse said client side executable code" (Specification, page 17, lines 13-16; Figure 11, item 1111), "so as to determine a subsequent state of the network application free of interaction with a user," (Specification, page 4, line 21 - page 5, lines 1; *see generally* Figure 6) "and said form merging instructions comprising: a reference to a form in a previously

downloaded web page," (Specification, page 16, lines 8-10) "said form in said previously downloaded web page being an HTML form;" (Specification, page 17, lines 9-11) "a reference to a form in said test script," (Specification, page 27, lines 12-16) "said form in said test script being a script form; (Specification, page 27, lines 12-16) "and instructions for merging said HTML form and said script form to produce a form to be submitted;" (Specification, page 27, line 19 bridging page 28, line 8) " and execute said context-full test script using said context-full page-level API for said replay engine." (Specification, page 30, lines 9-11; Specification, page 18, lines 5-9)

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1) Whether U.S. Patent No. 6,549,944 to Weinberg et al. ("Weinberg") discloses each and every element of claims 1-12 and 29 under 35 U.S.C. § 102(e).

2) Whether claims 13-28, 30 and 31 can properly be rejected as obvious under 35 U.S.C. § 103(a) based on the combination of Weinberg and U.S. Patent Publication No. 2002/0062342 to Sidles ("Sidles").

VII. ARGUMENT

Grounds of Rejection No. 1

Appellants submit that Weinberg is directed to "[a] visual Web site analysis program" for the "analysis, management and load testing of Web sites." (Weinberg, Abstract.) In accordance with Weinberg, a "tool operates by capturing actual HTTP and/or HTTPS (Secure HTTP) traffic between a standard web browser and the web server, and recording this traffic into a script file." (Weinberg, col. 2, lines 32-37.) The "Dynamic Scan™ feature" of the tool can "automatically append dynamically generated Web pages (such as pages generated using CGI scripts) to [the web site's] maps." (Weinberg, col. 8, lines 2-5.) "[The dynamic page scanning] process involves capturing the output of a Web browser when the user submits an HTML-embedded form (such as when the user submits a database query), and then reusing the captured dataset during the scanning process to recreate the form submission." (Weinberg, col. 9, lines 30-37.) The tool can then automatically generate "test scenarios (including test scripts) based on information stored within a

server access log file." (Weinberg, col. 3, lines 7-10.) "[T]he Web scripts are generated by capturing and recording the output of a standard Web browser during interactive browsing of the site by a user." (Weinberg, page 32, lines 37-40.) The scenarios can then be used to load test the web site. (Weinberg, col. 3, lines 34-36.)

Execution of a "web script," as described in Weinberg, is not the execution client-side executable code. Rather, the replay (i.e., execution) of a web script is merely a repetition of previously requested and recorded URLs. This replay is different than the execution of client-side executable code (e.g., JavaScript) to dynamically generate new (i.e., not previously recorded) URL requests, as discussed in more detail below.

In contrast, in the present invention, (e.g., independent claims 1 and 29), a system and a device are provided for automatic context management for testing, monitoring and automating a network application having "client side executable code." With reference to claim 1, the present invention recites "a tool operable to parse said client side executable code so as to determine a subsequent state of the network application free of interaction with a user." Independent claim 29 recites similar features.

The term "client side code execution" refers to software code (i.e., program instruction) that is executed within the web browser by the client (i.e., on the client computer). (Specification, page 11, lines 1-11.) Examples of client side executable code include JavaScript code embedded in HTML documents or in separate JavaScript documents, Java applets, VBScript, ActiveX controls, or other browser plug-ins. (Specification, page 11, lines 1-11.) "Code executed on the client side may dynamically assemble URLs and forms and cause page transitions using these URLs and forms." (Specification, page 11, lines 4-7.) "Such page transitions cannot be modeled by context-full replay instructions within a traditional page-level replay, because the URLs and forms may not correspond to any hyperlink or form contained in any previously downloaded web page." (Specification, page 11, lines 4-7.) Thus, "[c]lient-side code causes actions that are performed by the client, rather than by executing web application code on the server." (Specification, page 12, lines 17-18.)

Thus, the present invention is capable of performing "context management for testing, monitoring and automating a network application" of web sites that includes more than mere HTML web pages and HTML forms that are generated by the web server. Rather, the present invention can parse, record and replay test scripts that include web pages generated, in part, by client-side code. In contrast, Weinberg is limited to parsing standard HTML and HTML forms, completing those forms with previously collected data, and submitting the completed form as a standard HTTP request to the server. (Weinberg, col. 9, lines 30-37.)

The Examiner relies on Weinberg, column 23, line 15 bridging column 26, line 19, (especially col. 24, lines 1-33) as disclosing a "Dynamic Scan feature for scanning dynamic HTML code i.e., client side executable code, which may be set to automatically fill in dynamic forms without interaction from a user." (July 14, 2006 Final Office Action, item 10, page 5.) Appellants respectfully submit that the Examiner incorrectly interprets the Dynamic Scan feature of Weinberg.

Weinberg's dynamic scan feature is directed to the scanning and mapping of web pages that are **dynamically generated <u>on the server</u> (i.e., 'by a web site')**". (Weinberg, col. 23, lines 17-20 (emphasis added).) Weinberg discloses various tools for dynamically generating web pages on a server, including CGI, Microsoft's ISAPI, and Netscape's NSAPI, all of which are server based tools. (Weinberg, col. 23, lines 25-28.) However, Weinberg does not disclose scanning and mapping web pages that are **dynamically generated <u>on the client</u> (i.e., by the web browser)**, as claimed by the present invention. (*See* claims 1, 29, 30, and 31 reciting "network application having client side executable code"). Thus, the Examiner improperly interprets Weinberg's discussion concerning web pages that are dynamically generated by the server as disclosure of client-side executable code and web pages that are generated by such code. There is simply no mention of client side executable code in Weinberg.

In response to similar arguments presented by the Appellants in the October 16, 2006 "Amendment in Response to Final Office Action," the Examiner stated that "Weinberg explicitly teaches an automated testing tool for testing and monitoring web applications having client side executable code," (November 14, 2006, Advisory Action (*citing* Weinberg, col. 2, line 30 bridging

Col. 3, line 37; col. 8, lines 40-67)) and further cites column 24, lines 25-34 of Weinberg as explicitly disclosing "scanning and parsing client side executable code." However, the passages cited by the Examiner merely disclose a "dynamic page," which Weinberg defines as "a page that is generated 'on-the-fly' **by a Web site** in response to some user input, such as a database query." (Weinberg, col. 23, lines 17-20 (emphasis added).) Weinberg does not disclose scanning and parsing web pages that are dynamically generated on the client or client-side executable code. Further, the replay of "web script," as described in Weinberg, is not the execution client-side executable code. Thus, Weinberg does not disclose each and every element of claims 1 and 29.

Claims 2-12 depend from claim 1. These dependent claims are patentable over Weinberg for at least the same reasons as their base claim, i.e., claim 1. Therefore, for the reasons stated above, Appellants respectfully submit that Weinberg does not disclose each and every feature of claims 1-12 and 29.

Grounds of Rejection No. 2

Appellants submit that Sidles is directed to an automated form filling system that compares the field names of the form with the dictionary list of field names to see if it can find a match. If there is no exact match, fuzzy logic tries to "guess" how to fill the remaining data fields of the form (Sidles, paragraph 0040.) If the dictionary and fuzzy logic are unable to complete all the fields of the form, then the system looks in the history database to see if the specific site and form have been previously filled by a human person. (Sidles, paragraph 0041.) "[I]f the same form has been encountered and filled out previously . . . the form is filled out with user information using the previously entered information as a guide." (Sidles, paragraph 0018.)

Claims 13-27 depend from claim 1. Additionally, independent claims 30 and 31 recite features similar to those discussed above with respect to claim 1. Appellants submit that Sidles does not disclose or suggest the features demonstrated above to be missing from Weinberg with respect to claim 1. Therefore, for the reasons stated above, Appellants respectfully submit that the Examiner has failed to make out a prima facie case of obviousness with respect to claims 13-27, 30 and 31.

With respect to claim 28, the Examiner contends that "Sidles teaches that the comparison and generating steps are performed for each form in the session history, since all forms from the previous sessions are stored in the database for comparison against the submitted form." (July 14, 2006, Office Action, item 12, page 15.) The Examiner further states in the November 14, 2006 Advisory Action "Sidles explicitly teaches the application of rules to associate data with form field labels, **to generate data based upon differences** resulting from the comparing step, (Sidles, p. 6, par. 61-62 (emphasis added)) and therefore does not merely teach an exact match."

Appellants submit that, as cited by the Examiner, Sidles applies "rules that are used to guide the filling out of forms." (Sidles, ¶ [0061].) For each field of a form, Sidles applies the various rules and "associates numerous different fillable form field labels with different [] data base data values." (Sidles, ¶ [0062].) If multiple rules are applicable and assign different values to the same field label, "these ambiguities are resolved by the artificial intelligence system's fuzzy logic." (Sidles, ¶ [0063].) Appellants submit, that while Sidles applies rules to determine "specific user information in the [] database 1100 that is to be inserted into that particular field," (Sidles, ¶ [0062]) Sidles does not generate data based differences between the particular field and the rule being applied.

In contrast, claim 28 "compar[es] a form to be submitted to at least one form in a session history," and "generat[es] data based upon differences resulting from the comparing step." The generating step of the present invention generates data regarding the comparison of forms which is used in the step of "choosing one of the forms in said session history having the greatest similarity to said form to be submitted based upon the generating step results." Sidles does not generate data based upon the differences of any comparison or rule application. Rather, Sidles merely attempts to apply rules.

Appellants further submit that, because Sidles lacks the generating step, Sidles also lacks "choosing one of the forms in said session history . . . based upon the generating step results," as recited in claim 28. Furthermore, Sidles consequently lacks "applying form merging instructions to said chosen session history form," as recited in claim 28.

For at least the reasons set forth above, Appellants submit that claim 28 is not obvious in view of the combination of Weinberg and Sidles.

For all of the reasons set forth above, the rejections of claims 1-31 should be reversed. Appellants respectfully request that the application be remanded to the Primary Examiner with an instruction to withdraw the rejections, and pass the case to allowance.

Respectfully submitted,

By 

Dated: June 18, 2007

Kevin J. Beach
Registration No.: 60,422
DARBY & DARBY P.C.
P.O. Box 5257
New York, New York 10150-5257
(212) 527-7700
(212) 527-7701 (Fax)
Attorneys/Agents For Appellants

APPENDIXES

CLAIMS APPENDIX

The following is a copy of the claims involved in the appeal:

1. A system for automatic context management for testing, monitoring and automating a network application having client side executable code, said system comprising:

a tool operable to parse said client side executable code so as to determine a subsequent state of the network application free of interaction with a user;

a recorder operable to record at least one context-full test script; and

a replay engine of said tool having a context-full API, said replay engine operable to execute said context-full test script.

2. The system according to claim 1, wherein said context-full API is based on a page-level API.

3. The system according to claim 2, wherein said context-full API based on a page-level API comprises an extensible document parser operable to determine at least one parser extension.

4. The system according to claim 3, wherein said at least one parser extension includes a replay instruction specifying at least one parser addition and parameters for said parser addition.

5. The system according to claim 4, wherein said parser addition includes a plug-in module for said extensible document parser.

6. The system according to claim 3, wherein said extensible document parser is capable of being extended with at least one parser extension.

7. The system according to claim 4, wherein said at least one parser addition for said parser extension is selected from a library of parser additions.

8. The system according to claim 7, wherein each of said parser additions of said library of parser additions implements a specific parsing algorithm.

9. The system according to claim 8, wherein each of said parser additions of said library of parser additions implements an algorithm for parsing hyperlinks from a HTML document.

10. The system according to claim 8, wherein each of said parser additions of said library of parser additions implements an algorithm for parsing forms from a HTML document.

11. The system according to claim 8, wherein each of said parser additions of said library of parser additions implements an algorithm for parsing embedded documents from a HTML document.

12. The system according to claim 9, wherein said algorithm parses hyperlinks by searching text between a left and a right boundary string.

13. The system according to claim 2, wherein said context-full API comprises form merging replay instructions.

14. The system according to claim 13, wherein said form merging instructions comprise:

a reference to a form in a previously downloaded web page, said form in said previously downloaded web page being an HTML form;

a reference to a form in said at least one context-full test script, said form in said at least one context-full test script being a script form; and

instructions for merging said HTML form and said at least one context-full script form to produce a form to be submitted.

15. The system according to claim 14, wherein:

said instructions for merging said HTML form and said script form comprise merging instructions for each individual form field of said HTML form and said script form; and

said merging instructions for each said individual form field comprise at least one member selected from the group consisting of:

an instruction to send a form field value obtained from said HTML form;

an instruction to send a form field value specified in said script form; and

an instruction to not send one of said form fields[[;]].

16. The system according to claim 15, wherein said merging instructions comprise an instruction to use an action URL in said test script instead of an action URL obtained from said HTML form for said form to be submitted.

17. The system according to claim 1, wherein said recorder is operable to record a page-level test script comprising parser extensions and form merging instructions.

18. The system according to claim 17, wherein said recorder is operable to track a session history by building representations of all web pages downloaded so far during a recording session.

19. The system according to claim 17, wherein:

said context-full API is based on a page-level API;

said context-full API based on a page-level API comprises an extensible document parser; and

said recorder is operable to utilize said extensible document parser to parse at least one HTML document.

20. The system according to claim 17, wherein said recorder is operable to automatically detect which of said parser extensions and said form merging instructions are needed in order to record a test script which will correctly use dynamic information during a script replay.

21. The system according to claim 20, wherein said recorder is operable to detect the need for recording at least one of said parser extensions by detecting that a context-less replay instruction is to be recorded otherwise.

22. The system according to claim 20, wherein:

said context-full API comprises an extensible document parser;

said extensible document parser comprises at least one parser extension which is a replay instruction specifying at least one parser addition and parameters for said parser addition;

said parser addition for said parser extension can be chosen from a library of parser additions;

each of said parser additions of said library of parser additions implements a specific parsing algorithm; and

said recorder is operable to detect which one of said parser extensions is to be recorded by querying each of said parser additions for suitable parameters.

23. The system according to claim 1, wherein said recorder is operable to record form merging instructions by performing fuzzy form detection.

24. The system according to claim 23, wherein said recorder is operable to perform fuzzy form detection by comparing a form being submitted to all forms in a session history, operable to choose a form from said session history which is most similar to said form being submitted; and further operable to record said form merging instructions so that said recorded form merging instructions applied to said form chosen from said session history result in a form identical to said form being submitted.

25. The system according to claim 1, wherein said replay engine is operable to execute said context-full test script, said context-full test script comprising parser extensions and form merging instructions.

26. The system according to claim 25, wherein said replay engine is operable to track a session history by building representations of all web pages downloaded so far during a replaying session.

27. The system according to claim 25, wherein:

said context-full API is based on a page-level API;

said context-full API based on a page-level API comprises an extensible document parser; and

said replay engine is operable to use said extensible document parser to parse at least one HTML document.

28. A method of fuzzy form detection, said method comprising the steps of:

comparing a form to be submitted to at least one form in a session history;

generating data based upon differences resulting from the comparing step;

performing said comparing and generating steps for each form in said session history;

choosing one of the forms in said session history having the greatest similarity to said form to be submitted based upon the generating step results; and

applying form merging instructions to said chosen session history form to obtain a resulting form that is substantially identical to said form to be submitted.

29. A device for automatic context management for testing, monitoring and automating a network application having client side executable code, said device comprising:

a processor; and

a memory storing processing instructions for controlling the processor, the processor operative with the processing instructions to:

record at least one context-full test script; and

execute said context-full test script using a context-full API for a replay engine of a tool, said tool operable to parse said client side executable code so as to determine a subsequent state of the network application free of interaction with a user.

30. A device for automatic context management for testing, monitoring and automating a network application having client side executable code, said device comprising:

a processor; and

a memory storing processing instructions for controlling the processor, the processor operative with the processing instructions to:

record at least one context-full test script;

determine at least one parser extension using an extensible document parser of a context-full page-level API for a replay engine of a tool, said tool operable to parse said client side executable code so as to determine a subsequent state of the network application free of interaction with a user;

include a replay instruction specifying at least one parser addition and parameters for said parser addition in said at least one parser extension;

select said parser addition for said parser extension from a library of parser additions;

implement a specific parsing algorithm using each of said parser additions of said library of parser additions;

implement an algorithm for parsing hyperlinks from a HTML document using said parser addition of said library of parser additions;

parse hyperlinks by searching text between a left and a right boundary string using said algorithm; and

execute said context-full test script using said context-full page-level API for said replay engine.

31. A device for automatic context management for testing, monitoring and automating a network application having client side executable code, said device comprising:

a processor; and

a memory storing processing instructions for controlling the processor, the processor operative with the processing instructions to:

record at least one context-full test script;

generate form merging replay instructions in a context-full page-level API for a replay engine of a tool, said tool operable to parse said client side executable code so as to determine a subsequent state of the network application free of interaction with a user, and said form merging instructions comprising: a reference to a form in a previously downloaded web page, said form in said previously downloaded web page being an HTML form; a reference to a form in said test script, said form in said test script being a script form; and instructions for merging said HTML form and said script form to produce a form to be submitted; and

execute said context-full test script using said context-full page-level API for said replay engine.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings for this matter.